Estructura de Datos y Algoritmos

Grado en Ingeniería del Software (Grupo F). Curso 2016-2017
Examen parcial de febrero **Tiempo: 3 horas**

Instrucciones

- Debéis entregar un fichero .cpp para cada ejercicio, nombrado ejerX.cpp siendo X el número del ejercicio.
- Al principio de cada fichero que entreguéis debe aparecer, en un comentario, vuestro nombre y apellidos, dni, puesto de laboratorio y usuario del juez si lo habéis usado. También debéis incluir unas líneas explicando qué habéis conseguido hacer y qué no.
- Todo lo que no sea código C++ (explicaciones, especificaciones, invariantes, etc.) debe ir en los propios ficheros .cpp en comentarios debidamente indicados.
- Si queréis utilizar el juez online, id a la mesa del profesor a recoger un usuario/password y entrad en http://exacrc/domjudge/team
- Las plantillas para poder probar vuestras funciones se obtienen pulsando en el icono del Escritorio "Publicacion docente ...", después en "Alumno recogida docente", y en el programa que se abre, abriendo en la parte derecha la carpeta EDA-F, arrastrando a hlocal (en la izquierda) el fichero PlantillasGrupoF.cpp.
- La entrega se realiza pulsando en el icono del escritorio "Examenes en Labs ...", y posteriormente, utilizando el programa que se abre, colocando los ficheros a entregar en la carpeta de vuestro puesto (en el lado derecho).

Ejercicio 1 [4 puntos]

Dado un vector de números enteros positivos, ordenado en orden estrictamente creciente, se pide diseñar un algoritmo eficiente que elimine todos los números impares, dejando sólo los pares, ordenados de forma creciente. El procedimiento modificará el vector de entrada y su tamaño, y no deberá utilizar ningún array auxiliar. Los parámetros por lo tanto serán de entrada/salida.
Se pide:

a) (1 punto) Especificar el problema.

b) (1,5 puntos) Diseñar e implementar un algoritmo que resuelva el problema.

c) (1 punto) Escribir un invariante y una función cota que permitan demostrar la corrección del algoritmo implementado.

d) (0,5 puntos) Justificar el coste del algoritmo.

Entrada/Salida para el corrector automático

La entrada que espera el corrector automático consta de una serie de casos de prueba y acabará cuando se introduzca un -1. Cada caso de prueba consta de dos líneas, en la primera se indica el número de elementos del vector. En la segunda se dan los elementos, ordenados en orden estrictamente creciente y separados por el carácter blanco. Se cumple que el número de elementos del vector es mayor o igual que cero y cada componente del vector es estrictamente mayor que cero.
Para cada caso de prueba se escribe una línea con los elementos que quedan en el vector después de eliminar los elementos impares dejando un espacio blanco entre cada par de elementos. Si en el vector no queda ningún elemento se deja una línea en blanco.

Entrada	**Salida**
5 3 5 6 8 10	6 8 10
4 2 4 5 7	2 4
5 1 3 5 7 9	
5 2 4 6 8 10	2 4 6 8 10
6 3 6 8 9 13 14	6 8 14

Ejercicio 2 [3 puntos]

Se tiene un vector con $k > 0$ valores enteros diferentes. Cada valor se encuentra repetido k_i veces, encontrándose consecutivos todos los valores iguales del vector. Se pide encontrar el número de valores distintos. El problema se debe resolver empleando la técnica de divide y vencerás. Si existen valores repetidos, el vector no debe recorrerse completo, evitando siempre que se pueda el acceso a los elementos repetidos.

Entrada/Salida para el corrector automático

La entrada que espera el corrector automático consta de una serie de casos de prueba y acabará cuando se introduzca un 0. Cada caso de prueba consta de dos líneas: en la primera se indica el número de elementos del vector y en la segunda los valores separados por un espacio.
Para cada caso de prueba se escribe el número de valores diferentes del vector en una línea.

Entrada	Salida
4 <intro> 3 3 1 1	2
3 <intro> 2 2 2	1
8 <intro> 6 6 3 3 8 8 1 1	4
10 <intro> 3 3 3 6 6 2 2 2 2 2	3
14 <intro> 7 7 7 7 7 4 4 4 4 9 9 2 2 2	4
20 <intro> 5 5 5 5 5 5 5 5 4 4 4 4 4 4 1 1 1 7 9 2	6
1 <intro> 5	1
6 <intro> 6 5 5 5 5 2	3
6 <intro> 8 7 6 5 4 3	6

Ejercicio 3 [3 puntos]

Laura quiere construir una torre con piezas de colores. En su juego de construcciones hay piezas azules, rojas y verdes, de cada una de las cuales tiene un determinado número disponible, respectivamente a, r y v. Quiere construir una torre que contenga $n >= 2$ piezas en total, cada una encima de la anterior. No le gusta el color verde, así que nunca coloca dos piezas verdes juntas, ni permite que el número de piezas verdes supere al de piezas azules en ningún momento mientras se va construyendo la torre. Además, como el color rojo es su favorito, las torres que construye siempre tienen en la parte inferior una pieza roja, y en la torre final el número de piezas rojas debe ser mayor que la suma de las piezas azules y verdes.

Implementar un algoritmo que muestre todas las formas posibles que tiene de construir una torre de la altura deseada cumpliendo con las restricciones mencionadas.

Entrada/Salida para el corrector automático

La entrada que espera el corrector automático consta de una serie de casos de prueba y acabará cuando se introduzca una línea con cuatro ceros. Cada caso de prueba se escribe en una línea y consta de 4 enteros separados por blancos. El primero es mayor que uno y representa la altura de la torre. Los tres siguientes son mayores o iguales que cero y representan los cubos de color azul, rojo y verde respectivamente.
Para cada caso de prueba se escriben todas las posibles torres, una en cada línea ordenadas por orden lexicográfico y separando cada par de colores por un espacio. Cada caso termina con una línea en blanco. Si no se puede construir la torre con los números de bloques dados se escribirá *SIN SOLUCION*.

Entrada	Salida
4 4 4 4	rojo azul rojo rojo rojo rojo azul rojo rojo rojo rojo azul rojo rojo rojo rojo
5 2 2 2	SIN SOLUCION
5 3 3 1	rojo azul azul rojo rojo rojo azul rojo azul rojo rojo azul rojo rojo azul rojo azul rojo rojo verde rojo azul rojo verde rojo rojo azul verde rojo rojo rojo rojo azul azul rojo rojo rojo azul rojo azul rojo rojo azul rojo verde rojo rojo azul verde rojo rojo rojo rojo azul azul rojo rojo rojo azul verde
2 1 2 1	rojo rojo